CAD-Based Wafer Scanning Simulations Applicable to the Field of Semiconductor Device Manufacturing

Nikolay Bratovanov[1, a)]

[1]*Robotics Lab, Faculty of Automation, Technical University of Sofia, Bulgaria*
[a)]*nbratovanov@tu-sofia.bg*

Abstract. This paper contributes to the development of wafer scanning simulation algorithms, which have been successfully integrated into an existing substrate handling robot simulator based on computer-aided design (CAD) software. The scanning process, which is extremely important to the field of semiconductor device manufacturing and cleanroom robotics has to be described, analyzed, mathematically modelled and later implemented into the source code of the mentioned simulator, which has been written in Visual Basic .NET programming language. As a result, the end-used is allowed to efficiently perform wafer scanning simulations in a virtual environment, without depending on the physical equipment, and be able to retrieve precise information related to the particular automated system. The proposed approach is verified and evaluated with the help of an experimental wafer scanning simulation scenario, based on SolidWorks and its application-programming interface (API).

INTRODUCTION

Semiconductor device manufacturing is a complex and sophisticated industrial field, which has a crucial impact on the technological progress in global terms [1], [2], [3]. Employing a high level of automation imposed by a number of considerations (cleanliness, throughput requirements, precision, reliability etc.), the chip production industry is strongly dependent on a specific class of manipulators, known as wafer handling robots [4], [5]. In addition to their primary task of transporting the silicon wafers between the different processing tools (where various photolithographic and chemical processes are taking place) in a swift, smooth and safe manner, the wafer handling robots are also responsible for the execution of various auxiliary procedures, which are of no less importance to the overall automated tool performance.

Such a procedure, known as wafer scanning (also referred to as 'mapping'), is related to the process of inspecting the pockets of a given wafer carrier for the presence and proper placement of the wafers that are contained in it – normally inserted, cross-slotted or protruding [6]. The scanning operation starts at the front of the carrier, with the scanning sensors facing the wafers from a position known as the "Start Scanning Position" (which is a subject to calibration). In order to successfully fulfil the task, the robot's end-effector must be equipped with a sensor, which takes the corresponding measurements as the arm performs the scanning motion in front of the cassette (Fig. 1). During the scanning process, the scanner moves on a predefined trajectory. Depending on the scanning mode, this trajectory may be of two types:

- The scanning end-effector moves up and down in a straight line in front of the cassette (Fast Scanning).
- The scanning end-effector moves up in steps, and has the option of jabbing as well as sweeping each pocket (Slow Scanning).

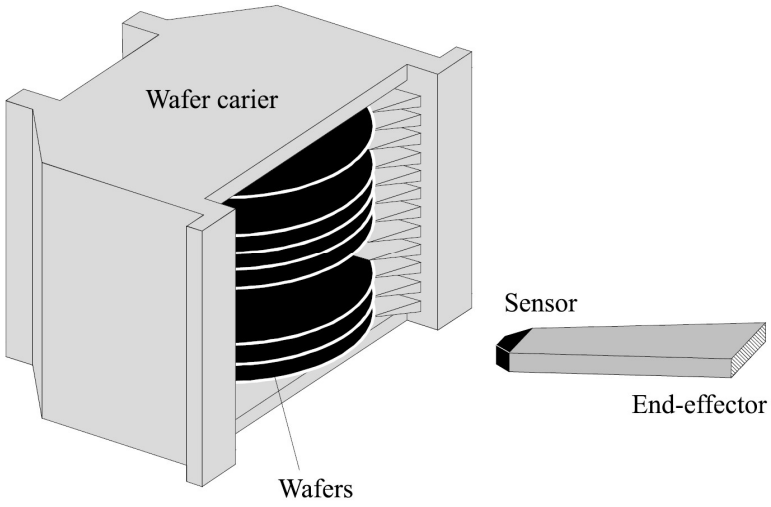

FIGURE 1. *Schematical representation of the wafer scanning process.*

The importance of the wafer scanning procedure for the semiconductor fabs (associated with detecting the presence or absence and slotting errors of silicon wafers in the processing equipment) combined with the already quite popular, customizable, efficient and precise 3D CAD-based software tools, which can be effectively utilized for robot motion simulation and offline programming purposes, motivate the implementation of the current task. The goal is enhancing the functionality of an already existing substrate-handling robotics simulator based on SolidWorks and its application-programming interface (API), which has been proposed and developed by the same author [7].

MODELING AND MATHEMATICAL REPRESENTATION OF THE WAFER SCANNING PROCESS

Considering the multibody nature of the wafer scanning procedure, its various calibration parameters, as well as specific trajectory patterns, it is obvious that numerous spatial relationships between all the involved objects (wafers, end-effectors, scanners, etc.) must be established in order to mathematically describe, model, implement and simulate the process in the SolidWorks API-based virtual environment. The formal representation of the wafer scanning procedure begins with reviewing a basic geometry task that introduces the two main components necessary for the modeling – a wafer and a scanner (through beam type).

"Wafer-laser beam" Intersection Check

Let us introduce two vectors for the purpose of representing the wafer with respect to a reference (base) coordinate frame – $\mathbf{r}_c$ (wafer's center) and $\mathbf{n}$ (wafer's normal vector). Similarly, the scanner's origin (coincident with the point of emitting the laser beam) with respect to the same base frame $O_b\mathbf{e}_{b1}\mathbf{e}_{b2}\mathbf{e}_{b3}$ is described by the vector $\mathbf{r}_0$, and the scanner's laser beam – by the unit vector $\mathbf{e}$, as shown in Fig. 2. It is obvious from the same figure that:

$$\mathbf{r}_{0c} = \mathbf{r}_c - \mathbf{r}_0 \tag{1}$$

In addition, two possible scenarios for the wafer's inclination (characterized by its normal vector $\mathbf{n}$) are considered at this point – the wafer is horizontally aligned (vector $\mathbf{n}$ is parallel to the unit vector $\mathbf{e}_{b3}$) or the wafer is tilted ($\mathbf{n}$ and $\mathbf{e}_{b3}$ are not parallel). These two scenarios correspond to the real-world situation where the wafer is either normally inserted, or cross-slotted inside the carrier. Depending on the specific case, the scalar s, which plays an important role for determining whether a wafer is detected at the slot being scanned or not, can be expressed as:

$$s = (\mathbf{r}_c - \mathbf{r}_0).\mathbf{n}/(\mathbf{e}.\mathbf{n}) \tag{2}$$

If the wafer is normally inserted, (2) can be rewritten as:

$$s = \mathbf{r}_{0c}.\,\mathbf{e} \tag{3}$$

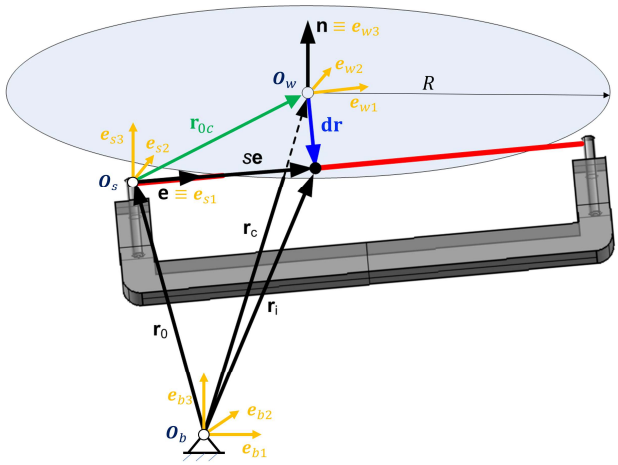

FIGURE 2. *Graphical representation of the 'wafer–scanner' relationship.*

The successful implementation of the wafer-detection algorithm, which relies on comparing the vector **dr** with the wafer's radius R, requires the derivation of the following additional relationships, all of them obvious from Fig. 2:

$$\mathbf{r}_i = \mathbf{r}_0 + s\mathbf{e} \tag{4}$$

$$\mathbf{dr} = \mathbf{r}_i - \mathbf{r}_c \tag{5}$$

Finally, the norm of the vector **dr** must be computed and compared with the radius R of the scanned wafer. It can be summarized that if $\|\mathbf{dr}\| < R$, a wafer is present at the particular slot, and if $\|\mathbf{dr}\| > R$, the slot is considered empty.

The presented concept requires the definition of several additional parameters, which are necessary for modeling the vertical motion (Z-axis) of the end-effector during wafer scanning, and reflecting the scanner's ability to derive (latch) the particular Z-coordinate at which a wafer has been detected – e.g. initial/final scanning vertical positions, scanning Z-step, wafer thickness, etc. (Fig. 3).

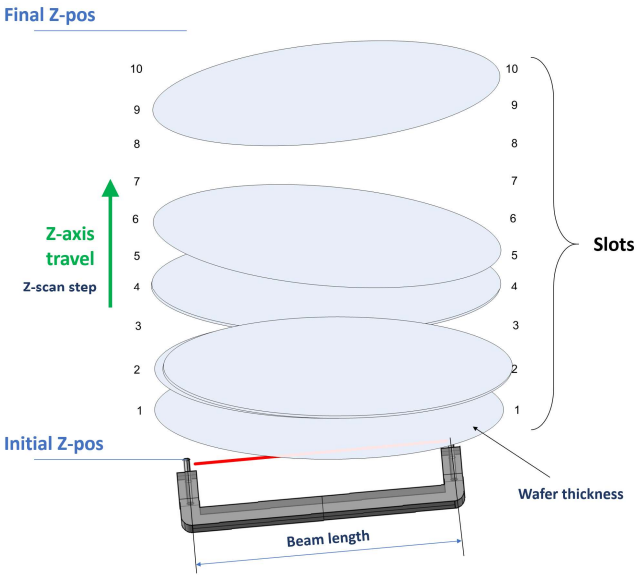

FIGURE 3. *Additional scanning parameters associated with the scanner's vertical travel.*

In other words, once triggered, the scanning algorithm must be continuously performed as the robot's arm elevation changes (based on a predefined step) – from the initial Z-position (the beginning of the scanning process) until the final Z-position has been reached. Thus, the whole wafer carrier gets "covered" by the scanner and complete scanning information associated with all slots is generated and output by the system.

"Wafer-scanner" Accessibility Check

Considering that not a small number of real-time calculations have to be performed for each available wafer, it is reasonable to take into account the problem of computational resources and overhead. In this regard, it is quite logical to assume that any wafers that are outside the scanner's working range should be excluded from the "wafer-laser beam" intersection algorithm, presented in the previous section, prior to its execution. For example, if there are multiple cassettes at the particular automated tool being analyzed, and it is the desire to simulate a scanning procedure of the first one, it is more than obvious that the laser beam of the scanner will not interfere with the wafers from the second cassette, hence all the associated vector calculations become irrelevant.

For this reason, the so-called "wafer-scanner" accessibility check must be implemented and performed for each available wafer. If it is determined that the wafer is outside the scanner's working range, the "intersect" algorithm is not executed, and the procedure continues with analyzing the next wafer. The principle of this check is once again based on straightforward vector operations, which resolve how far-off the wafer is from the scanner both in the plane (XY), and in vertical direction (Z).

IMPLEMENTATION OF THE WAFER SCANNING PROCESS TO AN EXISTING CAD-BASED ROBOT SIMULATOR

As already mentioned, one of the main reasons for developing the mathematical relationships and formal descriptions presented above is related to the integration and efficient utilization of the wafer scanning process with an already available CAD-based simulator, specially designed for motion simulations and offline programming of substrate handling robots.

Based on the popular 3D CAD software SolidWorks and its API, the simulator allows the real-time execution of motion commands, thanks to the established connection with the robots' motion control software (MCS). This type of functionality is associated with several considerations and development stages, e.g. proper coordinate frame attachment, kinematic modeling of the robot arm, implementation of a continuous text file reading procedure, collision detection algorithms, wafer handling simulations, etc. [8]. In order to successfully integrate the wafer scanning process, however, several important steps, reviewed in the following subsections, must be performed.

3D modeling of All Wafer Scanning-Related Components

The first, and quite obvious step is oriented towards the introduction of 3D models of all the components, related to the wafer scanning process – carriers, wafers, and the scanner itself. While these models are most likely available (already created for design and manufacturing purposes), the chances are that their coordinate frames are randomly located. Hence, the most important stage of the 3D modeling step is to make sure that all the frames are attached to the components in the same manner as described/shown in Fig. 2 (see $O_s\mathbf{e}_{s1}\mathbf{e}_{s2}\mathbf{e}_{s3}$ and $O_w\mathbf{e}_{w1}\mathbf{e}_{w2}\mathbf{e}_{w3}$), in order to correspond to the mathematical expressions that are later going to be implemented into the VB.NET code of the simulator (Fig. 4).

VB.NET Implementation of the Wafer Scanning Algorithms

Once the mentioned 3D models are successfully introduced into the overall robot simulation assembly file, they must be reflected into the VB.NET-based code of the existing simulator by implementing the two wafer scanning algorithms, described in the previous sections. Being a straightforward procedure, one of the most important step is associated with obtaining the necessary geometrical information from the native SolidWorks environment (frame origins, normal vectors, wafer thickness, laser beam length, etc.), and assign the retrieved values to the corresponding variables of the VB.NET scanning module.

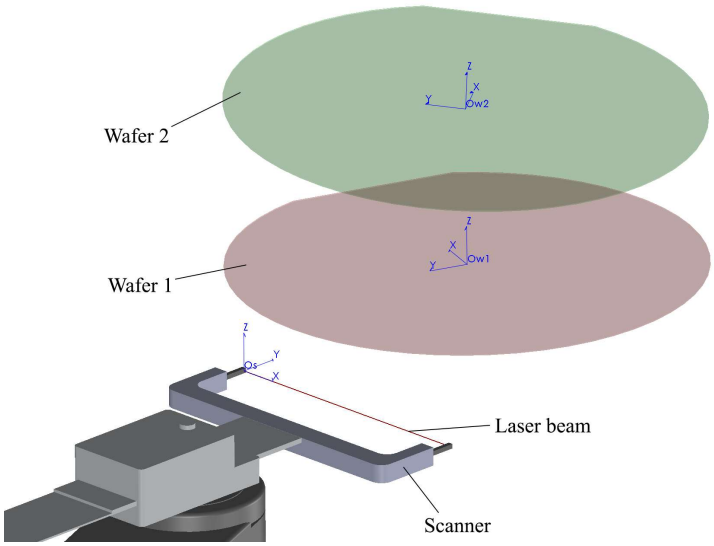

FIGURE 4. *SolidWorks 3D modeling of the wafer scanning-related components.*

This is done with the help of specific SolidWorks API properties, such as *Transform2*, *ArrayData*, *SensorValue*, etc. [9]. The two wafer scanning algorithms are then developed as separate procedures and are introduced into the main VB.NET method of the simulator, making sure that they are called at the right moment of the code execution.

In order to achieve the desired functionality, the following sequence of actions is to be established:

1. The SolidWorks simulator is active (real-time 'MCS-SolidWorks' connection is established).
2. Geometrical information for all scanning-related components is being continuously obtained.
3. The command status of the MCS is being continuously checked by the SolidWorks simulator.
4. If a scanning command is detected, the simulator performs the corresponding wafer-scanning checks consequently for each wafer available in the 3D assembly, as the arm's vertical (Z) travel changes with the specified Z-step:
    a. *"Wafer-scanner"* accessibility check (returns *True* or *False*).
    b. *"Wafer-laser beam"* intersection check (if a. returned *True*).
5. The retrieved scanning data is stored and sent to the MCS.

Development of I/O Procedures Facilitating the Wafer Scanning Simulation

In addition to implementation of the main wafer scanning algorithms, it becomes obvious from the above sequence (namely steps 3 and 5) that several I/O procedures are needed in order to facilitate the overall wafer scanning simulation. These procedures, based on text file reading/writing methods, are responsible for the execution of the following tasks:

- Obtaining/reading wafer scanning-related values, specified by the user (e.g. vertical (Z) travel of the arm, scanning Z-step, calibration parameters, etc.) – '*parameters.txt*'.
- Monitoring/reading the wafer scanning command status (which is exported to a specific text file, generated by the robot MCS) – '*commands.txt*'.
- Storing/writing the scanning data after each wafer scanning simulation is over (which is later accessed by the robot MCS) – '*results.txt*'.

As a result of the reviewed steps, the wafer scanning procedure, modeled and formally described in the second section, has been successfully implemented and integrated into an existing substrate handling robot simulator, based on the 3D CAD software SolidWorks and its application-programming interface (API), thus extending its overall functionality and efficiency. The proposed approach is verified and evaluated with the help of an experimental wafer scanning simulation scenario, presented in the following chapter.

EXPERIMENTAL SIMULATIONS AND VERIFICATION OF THE PROPOSED APPROACH

In order to verify the functionality, accuracy and effectiveness of the implemented algorithms for wafer scanning simulations, an exemplary experimental scenario has been set-up, executed and reviewed. The utilized SolidWorks API-based simulation layout, shown in Fig. 5, is comprised of a substrate handling robot (GB8Y) and a load port device, where a 300mm wafer carrier (referred to as FOUP) is located.

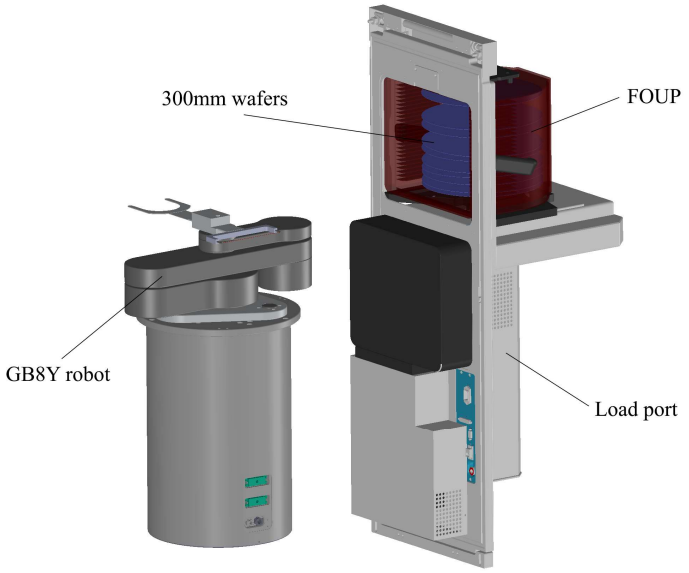

FIGURE 5. *SolidWorks API-based wafer scanning simulation layout.*

As it can also be seen from the figure, the FOUP (which has 25 slots) contains multiple wafers, whose arrangement is given in Fig. 6. All the possible wafer situations – normally inserted, absent or cross-slotted – are reflected. Considering the provided information, the purpose of the experimental scenario is to confirm the functionality and verify the accuracy of the wafer scanning algorithms that have been adapted and implemented to the particular layout, by executing a test scanning simulation and comparing the retrieved scanning results to the actual wafer placement-related data, shown in Fig. 6.

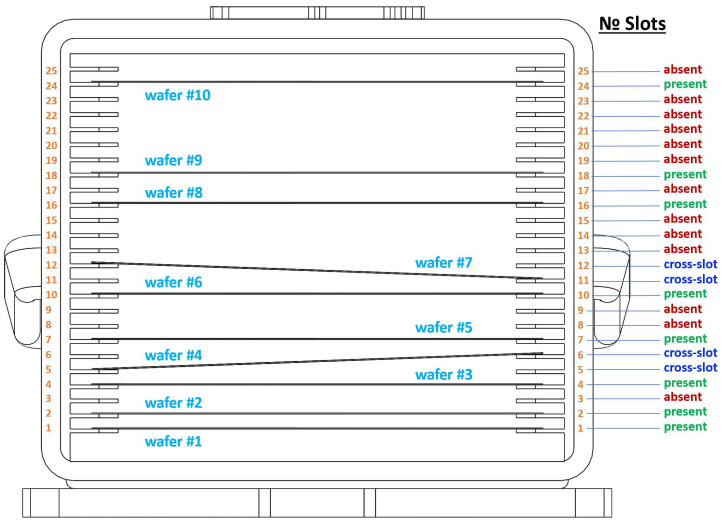

FIGURE 6. *Wafer placement-related data used for the experimental simulations.*

Before the wafer scanning simulation takes place, two parameters must be defined by the user – the final Z-position and the scanning Z-step (given in mm). Based on them and considering the initial Z-position of the scanner (taken from the robot's direct kinematics module), the simulator calculates the Z-axis travel that must be executed during the scanning process (which, obviously, must be greater than the height of the carrier being scanned) – 270mm in the current case. The step is set to 0.05mm. In addition, the robot arm configuration, related to the scanning initial position must also be taught.

The final stage of the experimental wafer scanning simulation deals with switching the simulator into its active mode (establish a connection to the robot's MCS) and initiating the wafer scanning process by executing the corresponding command (e.g. 'SCF') into the motion control software. A series of actions is triggered as a result:

1. The robot's motion, associated with specific scanning trajectories, is visualized inside the native SolidWorks environment.
2. A text file containing all the wafer intersection Z-coordinates, where the laser beam "enters" (IN-position) and "leaves" (OUT-position) each wafer in the Z-vertical direction, is being generated (entitled '*results.txt*').
3. The MCS receives the corresponding scanning data, based on the '*results.txt*' information, in the form of a string of zeroes, ones and twos, which shows the status of each slot of the carrier in the following fashion: **0** – slot is empty; **1** – wafer is present; **2** – wafer is cross-slotted.

Knowing the coordinates of each carrier's slot (as well as the thickness of all wafers = 0.7mm) and interpreting the information from the '*results.txt*' file given in table 1, the wafer scanning algorithms implemented into the SolidWorks API-based robot simulator are able to determine the final scanning data code, which corresponds to the actual wafer placement, already visualized in Fig. 6.

TABLE 1. *Summarized data associated with the performed wafer scanning simulation.*

Slot #	Slot Z-coord [mm]	Wafer #	Scan IN-pos [mm]	Scan OUT-pos [mm]	Diff [mm]
1	293.94	1	293.96	294.62	0.66
2	303.94	2	303.97	304.63	0.66
3	313.94	absent			
4	323.94	3	323.99	324.65	0.66
5	333.94	4	336.53	341.92	5.38
6	343.94				
7	353.94	5	353.96	354.62	0.66
8	363.94	absent			
9	373.94				
10	383.94	6	383.98	384.64	0.66
11	393.94	7	396.58	401.96	5.38
12	403.94				
13	413.94	absent			
14	423.94				
15	433.94				
16	443.94	8	443.98	444.64	0.66
17	453.94	absent			
18	463.94	9	463.99	464.65	0.66
19	473.94	absent			
20	483.94				
21	493.94				
22	503.94				
23	513.94				
24	523.94	10	523.99	524.65	0.66
25	533.94	absent			

It is confirmed by Fig. 7 that final scanning code, which is eventually sent to the robot MCS, allowing its further utilization and integration into the wafer-handling automation process, has the following structure: **1101221001220001010000010**. In addition, analysis and manual evaluation of the intersection Z-coordinates using the dedicated SolidWorks 'Measure' tool confirms the high level of accuracy and efficiency of the proposed approach for wafer scanning simulations, thus turning it into a reliable tool and enhancing the overall functionality of the SolidWorks API-based simulator for substrate handling robots.

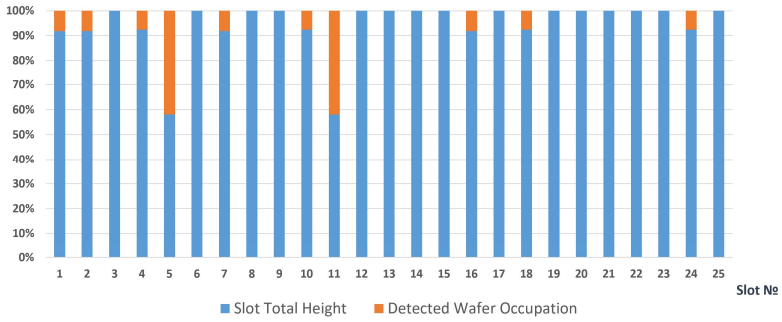

FIGURE 7. *Graphical representation of the retrieved scanning data.*

CONCLUSIONS AND FUTURE WORK

The paper contributes to the development and implementation of wafer scanning simulation algorithms, which have been successfully adapted and integrated into an existing substrate handling robot simulator, based on SolidWorks API. As a result of the proposed modeling and mathematical representation of the wafer scanning process, the simulator's functionality has been considerably extended, turning it into an even more precise, versatile and effective tool for the execution of complete motion simulations, offline programming, customer layout analysis, collision detection and other vital for the semiconductor device manufacturing automation tasks. The implemented algorithms facilitate the execution of real-time wafer scanning simulations, reflecting the specific wafer arrangement, sensor features and geometry of the tool being analyzed.

Some of the future developments related to the presented project, are oriented towards improving the wafer scanning simulation procedures by taking into account all possible wafer placement situations (i.e. two wafers in one and the same slot, protruding wafers), simulation of additional scanner types, flipping end-effectors, etc., as well as enhancing the currently available wafer handling simulation algorithms, implemented into the SolidWorks API-based substrate handling robot simulator.

ACKNOWLEDGEMENTS

This work is supported by the Technical University of Sofia, Bulgaria under Research Project of the Ministry of Education and Science "Young scientists and postdoctoral fellows"/2021.

REFERENCES

1. W. Den, S. Hu, C. Garza and O. Zargar, "Review-Airborne Molecular Contamination: Recent Developments in the Understanding and Minimization for Advanced Semiconductor Device Manufacturing," in *ECS Journal of Solid State Science and Technology*, **9**(6), 064003 (2020).
2. A. Bhvanagarwala, S. Borkar, T. Sakurai and S. Narendra S, "EP2: The semiconductor industry in 2025," in *2010 IEEE International Solid-State Circuits Conference*, pp. 534–535 (2010).
3. I. Kao and C. Chung C, *Wafer Manufacturing: Shaping of Single Crystal Silicon Wafers*, ISBN: 978-1-118-69623-1 (2021).
4. M. Cong and D. Cui, "Wafer-Handling Robots and Applications," in *Recent Patents on Engineering*, **3**(3), pp. 170–177 (2009).
5. H. Cheng, H. Chen and B. Mooring, "Accuracy Analysis of Dynamic-Wafer-Handling Robotic System in Semiconductor Manufacturing," *IEEE Transactions on Industrial Electronics*, **61**(3), pp. 1402–1410 (2014).
6. "Introduction to the wafer scanning process," *Gencobot GPR Series Reference Manual*, p. 117 (2002).
7. N. Bratovanov, "Robot Modeling, Motion Simulation and Off-line Programming Based on SolidWorks API," in *Third IEEE International Conference on Robotic Computing (IRC)*, pp. 574–579 (2019).
8. N. Bratovanov, "SolidWorks Add-In for Motion Simulation, Layout Analysis and Collision Detection of Substrate Handling Robots," *Proceedings of Technical University of Sofia*, ISSN 1311-0829 **70**(1), pp. 29–39 (2020).
9. L. Malpass, *SolidWorks API Series 1: Programming & Automation,* Second edition, p. 16 (2011).